Automated Theorem Proving

Lecture 6

Cezary Kaliszyk

June 23, 2020

Outline

Previous Lecture

- Equality
- Paramodulation
- Superposition

Today

- Current research topics in automated theorem proving
- Exam start

Higher-order logic and ATPs

What is "higher-order"

Terms are not just applications of variables and constants, but can also contain abstractions (full λ-terms).

Why is this interesting

Useful for formalization of mathematics and for proof about higher-order programs

Recent development

A competition exists since about 10 years but many current provers (E, Vampire) try to extend their superposition to higher-order logic only in the last 2 years (E, Vampire)

Reasoning in higher-order logic (1/2)

Why is it not straightforward?

Consider an ordering on higher-order terms. Without loss of generality:

$$a \qquad < \qquad b$$

For superposition the ordering should be closed under context, so it should still be:

$$(\lambda x.c)a \qquad < \qquad (\lambda x.c)b$$

Lets β-reduce our terms:

$$c \qquad < \qquad c$$

So an ordering compatible with superposition, that is having the necessary properties cannot exist...

Reasoning in higher-order logic (2/2)

Instead various restrictions of higher-order logic are used:

- Lambda-free higher-order logic [*Blanchette et al*]
- Proposition-free higher-order logic
- Higher-order logic restricted to induction [*Voronkov et al*]

Alternatives to superposition

- Higher-order automated proofs done using tableaux
- Efficiency recovered by SAT integration in Satallax [*Brown et al*]

AI in Theorem Proving

With recent progress in machine learning

Emerging topic also for theorem proving

Many places where we can apply learning

In different calculi, for different choices, before the call to prover or integrated

AI theorem proving techniques (1/3)

AI theorem proving techniques in three levels

High-level AI guidance

- Problems where machine learning helps predict and suggest actions that a human normally selects
- First such problem is premise selection: given a statement that we are trying to prove and a very large set of axioms (for example all the mathematical books in existence), find the parts of the knowledge that are most useful. For example if we have a fact that looks like topology, a human can find books on topology in a library. A machine learning algorithm can process thousands of books and facts in them - can it select the most useful ones for our goal?
- For this task we need a sufficiently good characterization of the goal and the axiom. As such it is necessary to find suitable features and learning techniques. And gather sufficient learning data.
- We often have seconds to minutes to perform the actual prediction, with more the human that is to run the ATP will likely do better.

AI theorem proving techniques (2/3)

Mid-level AI guidance

- Problems where the learning can select a strategy for an automated prover, for example the ordering and its parameters in ordered resolution.
- Selecting lemmas that can be reused, proposing intermediate lemmas
- We often have up to a second seconds to perform the actual prediction, with more time running a tool for longer will likely do better.

AI theorem proving techniques (3/3)

Low-level AI guidance

- AI can also be used to directly guide the actions of an automated theorem prover
- This means selecting (almost) every inference step by previous knowledge
- Depending on a calculus this may mean hundreds to tens of thousands of inferences per second
- In some ATPs this may be the selection of a clause to expand, in some ATPs this may be the selection of substitutions to apply, in some cases a branch etc.
- In order to make very fast decisions it is necessary to have very good proof-state characterizations and fast relevance: typically minimal statistical methods
- Most recently also random forest predictions, despite slowing the prover down 2-5 times can allow having overall higher performance

Two AI heuristics common in ATPs

Recently a lot of research in all these levels.

In this course we will only give ideas of two heuristics common in ATPs

The first is SInE [*Hoder*]
It looks at symbols in statements to reduce the size of a large problem, heuristically getting rid of statements that will not be useful and will clutter the indexing structures.

SInE: original idea by Hoder

Basic algorithm

If symbol s is d-relevant and appears in axiom a, then a and all symbols in a become $d+1$-relevant.

Problem: Common Symbols

- Simple *relevance* usually selects all axioms
- Because of common symbols, such as `subclass` or `subsumes`

```
subclass (beverage, liquid).
subclass (chair, furniture).
```

Solution: Trigger based selection

"appears" is changed to "triggers"

But how to know if s is common?

Approximate by number of occurrences in the current problem

SInE: Tolerance

- Only symbols with t-times more occurrences than the least common symbol trigger an axiom
- For $t = \infty$ this is the same as relevance

1:	subclass(X,Y) ∧ subclass(Y,Z) → subclass(X,Z)
	subclass(petrol,liquid)
	¬subclass(stone,liquid)
2:	subclass(beverage,liquid)
1:	subclass(beer,beverage)
	subclass(guinness,beer)
	subclass(pilsner,beer)

Occ.	Symbols
7	subclass
3	liquid, beer
2	beverage
1	petrol, stone, guinness, pilsner

? subclass(beer,liquid)

[*Hoder*]

In the example, the right table lists symbol occurrences, the left table lists the rounds in which particular axioms are triggered. In two rounds exactly the needed axioms to prove the conjecture are triggered.

What about actually guiding an automated prover

Reminder: one of the most important decisions in the given-clause algorithm is the selection of the next clause to process

Are there heuristics that we can use there, that would be unfair but particularly efficient?

Maybe given enough data from successful (and not successful) proofs we can have some statistics as to the useful clauses to process?

A guided version of E prover: Data Collection

“MPTP” (Millions of problems for theorem proving) [*Urban 2006*]

- Is a set of 150,000 FOF statements
- That correspond to a mathematical proof library

32,521 theorems with ≥ 1 proof

- We can split them into 90% training and 10% testing
- We can see if we can do more of the testing problems having learned the training ones
- (We can take different parts of the set to cover all)

Collect all CNF intermediate steps

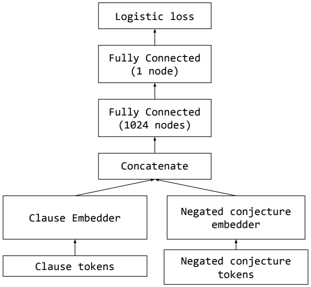
- and unprocessed clauses when a proof is found

Prediction Architectures

In the research either features and simpler machine learning was used

- e.g. group of J. Urban [*Jakubuv 2017, Chwalevsky 2018*]

Or neural networks (details not explained in this course) [*Loos et al 2017*]


Logistic loss
Fully Connected (1 node)
Fully Connected (1024 nodes)
Concatenate
Clause Embedder
Negated conjecture embedder
Clause tokens
Negated conjecture tokens

In both setups the purpose of the prediction is to give a rating to a new clause, so it is input in the E-prover unprocessed queue with that rating

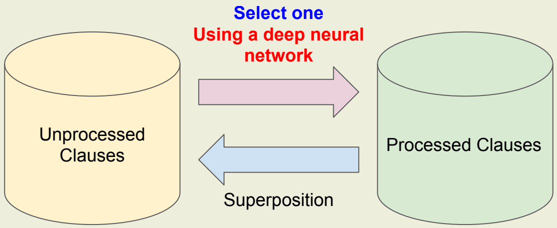
Model accuracy

Depending on the used dataset (strategy trained on, exact network architecture, ...) the accuracy was between 75 and 85 percent

Better than random, but somewhat unsatisfactory...

Improving Proof Search inside E

Overview



Problem

- Machine learning evaluation is slow
- In case of neural networks it is slower than combining selected clause with all processed clauses
- Various solutions tried, including deeper integration, batching, hybrid heuristics

AI in Eprover: results

(Note: last major evaluation in 2017, with many improvements since then)

AI-guided Eprover

Proved 7.4% of the statements not proved by any traditional heuristics

Additional Literature (not required)

Karel Chvalovský, Jan Jakubuv, Martin Suda, and Josef Urban.
ENIGMA-NG: efficient neural and gradient-boosted inference guidance for E.
In Pascal Fontaine, editor, *Automated Deduction - CADE 27 - 27th International Conference*, volume 11716 of *Lecture Notes in Computer Science*, pages 197–215. Springer, 2019.

Krystof Hoder and Andrei Voronkov.
Sine qua non for large theory reasoning.
In Nikolaj Bjørner and Viorica Sofronie-Stokkermans, editors, *Automated Deduction - CADE-23 - 23rd International Conference*, volume 6803 of *Lecture Notes in Computer Science*, pages 299–314. Springer, 2011.

Sarah M. Loos, Geoffrey Irving, Christian Szegedy, and Cezary Kaliszyk.
Deep network guided proof search.
In Thomas Eiter and David Sands, editors, *LPAR-21, 21st International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, volume 46 of *EPiC Series in Computing*, pages 85–105. EasyChair, 2017.

Final Exercises (or “Virtual Online Home Exam”) (1/2)

To be submitted before June 30

(1) Tableaux

Consider the CNF problem (to disprove), X, Y, Z are variables

$$(\neg P(X,Y) \vee P(Y,X)) \wedge (\neg P(X,Y) \vee \neg P(Y,Z) \vee P(X,Z)) \wedge P(X,f(X)) \wedge \neg P(X,X)$$

Find a closed connection tableaux for this problem

(2) Fairness

Consider the following clause selection strategy: Symbols present in the original TPTP conjecture are considered important, and symbols present in the axioms are considered unimportant. The priority of a clause is the difference between the number of important symbols present in it minus the number of unimportant symbols.

Is this strategy fair? If so, justify it. If not, give a counterexample.

continues on next page

Final Exercises (or "Virtual Online Home Exam") (2/2)

To be submitted before June 30

(3) Ordered resolution

Consider the ordering on predicates $a < b < c < ... < z$. Try to find a proof by ordered resolution with this order of:

```
cnf(2, axiom, ~a).
cnf(2, axiom, e | ~y).    cnf(2, axiom, z | ~c).
cnf(2, axiom, a | ~y).    cnf(2, axiom, y | ~c).
cnf(2, axiom, f | ~y).    cnf(2, axiom, w | ~c).
cnf(3, negated_conjecture, c).
```

(4) Prover Guidance

What are the most important heuristics that could be used to guide a tableaux prover? Justify your answer.

Send the answers to `cezary.kaliszyk@uibk.ac.at`